IN THE CLAIMS

Please amend the claims as follows:

1. (Withdrawn) A method for processing a memory instruction, the method comprising:
obtaining a memory instruction;
obtaining one or more memory address operands;
creating a virtual memory address using the one or more memory address operands;
translating the virtual memory address into a physical memory address; and
executing the memory instruction using a cache controller,
wherein the cache controller uses the memory instruction and the physical memory address to determine whether to access a portion of a local or a remote cache.

2. (Withdrawn) The method of claim 1, wherein the obtaining of the memory instruction includes obtaining a scalar memory instruction.

3. (Withdrawn) The method of claim 1, wherein the obtaining of the memory instruction includes obtaining a memory instruction selected from a group consisting of a scalar load instruction, a scalar store instruction, a prefetch instruction, a synchronization instruction, and an atomic memory operation (AMO) instruction.

4. (Withdrawn) The method of claim 1, wherein the obtaining of the memory instruction includes obtaining a scalar store instruction, and wherein the method further comprises obtaining scalar store data.

5. (Withdrawn) The method of claim 1, wherein the translating includes translating the virtual memory address into a physical memory address using a translation look-aside buffer (TLB).

6. (Withdrawn) The method of claim 1, wherein the method further comprises committing the memory instruction before execution.

7. (Currently Amended) A computerized method, comprising:

obtaining a memory request;

storing the memory request in an Initial Request Queue (IRQ); and

processing the memory request from the IRQ by a cache controller, wherein processing includes:

identifying a type of the memory request, ~~and~~

~~processing the memory request~~ determining whether the memory request hits in ~~both~~ a local cache, ~~and an~~

determining whether a portion of an address associated with the memory request matches one or more partial addresses in a Force Order Queue (FOQ),

~~wherein processing includes determining~~ if a portion of an address associated with the memory request matches one or more partial addresses in the FOQ, preventing the memory request from being satisfied in the local cache, and[,]

if the memory request misses in the cache and, at the same time, the address does not match one or more partial addresses in the FOQ, adding the memory request to the FOQ and allocating a cache line in the local cache corresponding to the local cache miss.

8. (Original) The computerized method of claim 7, wherein the obtaining of the memory request includes obtaining a memory load or a memory store request.

9-11. (Canceled)

12. (Currently Amended) The computerized method of claim 7, wherein ~~the processing of the memory request using the FOQ~~ preventing the memory request from being satisfied in the local cache includes processing the memory request using the FOQ when the memory request matches a corresponding request in the FOQ.

13. (Canceled)

14. (Currently Amended) The computerized method of claim 7, wherein ~~the processing of the memory request using the FOQ~~ determining whether a portion of an address associated with the memory request matches one or more partial addresses in an Force Order Queue (FOQ) includes processing the memory request in the FOQ when local cache processing is bypassed.

15. (Currently Amended) The computerized method of claim 7, wherein ~~the processing of the memory request using the FOQ~~ determining whether a portion of an address associated with the memory request matches one or more partial addresses in an Force Order Queue (FOQ) includes processing the memory request in the FOQ when the memory request includes a synchronization request that causes local cache processing to be bypassed.

16. (Withdrawn) A computerized method, comprising:

adding a set of memory instructions of a first type to a memory instruction container;

executing the set of memory instructions of the first type from the memory instruction container;

receiving a memory instruction of a second type;

receiving a synchronization instruction that temporarily blocks further execution of memory instructions of the first type from the memory instruction container;

adding an additional set of memory instructions of the first type to the memory instruction container;

executing the memory instruction of the second type; and

upon such execution of the memory instruction of the second type, executing the additional set of memory instructions of the first type from the memory instruction container.

17. (Withdrawn) The computerized method of claim 16, wherein the adding of the set of memory instructions of the first type to the memory instruction container includes adding the set of memory instructions of the first type to a memory instruction queue.

18. (Withdrawn) The computerized method of claim 16, wherein the adding of the set of memory instructions of the first type to the memory instruction container includes adding a set of scalar memory instructions to the memory instruction container.

19. (Withdrawn) The computerized method of claim 16, wherein the executing of the set of memory instructions of the first type from the memory instruction container includes accessing a local or remote cache unit.

20. (Withdrawn) The computerized method of claim 16, wherein the receiving of the memory instruction of the second type includes receiving a vector memory instruction.

21. (Withdrawn) The computerized method of claim 16, wherein the executing of the memory instruction of the second type includes accessing a remote cache unit, and updating a local cache unit with contents from the remote cache unit.

22. (Currently Amended) A scalar processor, comprising:

a cache;

an Initial Request Queue (IRQ); and

a cache controller having a Force Order Queue (FOQ);

wherein the IRQ buffers a scalar load/store command having a scalar load/store instruction and one or more addresses and sends the scalar load/store command to the cache controller and the cache;

wherein the cache services the scalar load/store command received from the IRQ when the scalar load/store command hits in the cache and a portion of one of the addresses of the one

or more addresses in the scalar load/store command does not match one or more partial addresses in the FOQ;

~~wherein the cache controller includes an Force Order Queue (FOQ),~~ wherein the scalar load/store command is added to the FOQ when the scalar load/store command misses in the cache; and

wherein one or more lines in the cache are allocated for cache line replacement when the scalar load/store command is added to the FOQ and the address for the cache line does not match a partial address in the FOQ.

23. (Canceled)

24. (Original) The scalar processor of claim 22, wherein the scalar load/store unit further includes an address generator to generate one or more physical addresses from the one or more addresses of the scalar load/store command.

25. (Original) The scalar processor of claim 24, wherein the address generator generates the one or more physical addresses using a translation look-aside buffer (TLB).

26. (Withdrawn) A processing unit, comprising:

a remote cache interface; and

a scalar processor having a local cache,

wherein the scalar processor dispatches a memory instruction, obtains one or more address operands, creates a virtual address from the one or more address operands, translates the virtual address into a physical address using a translation look-aside buffer (TLB), and executes the memory instruction using a cache controller, the cache controller determining whether to obtain data from the local cache or to allocate one or more lines of the local cache through the remote cache interface.

27. (Withdrawn) A processing unit, comprising:

a cache interface;

a vector processor; and

a scalar processor,

wherein the scalar processor processes a plurality of scalar memory instructions in a memory instruction container,

wherein the vector processor receives a vector memory instruction,

wherein the scalar processor receives a synchronization instruction and temporarily blocks further processing of scalar memory instructions in the memory instruction container,

wherein the vector processor processes the vector memory instruction by accessing a cache through the cache interface, and

wherein the scalar processor continues to process further scalar memory instructions in the memory instruction container after the vector processor has processed the vector memory instruction.

28. (Withdrawn) The processing unit of claim 27, wherein the memory instruction container comprises a memory instruction queue.

29. (Currently Amended) A scalar processor, comprising:

a cache;

an Initial Request Queue (IRQ); and

a plurality of cache controllers, wherein each cache controller includes a Force Order Queue (FOQ);

wherein the IRQ buffers a scalar load/store command having a scalar load/store instruction and one or more addresses and sends the scalar load/store command to one of the cache and to the cache controllers;

wherein the cache services the scalar load/store command received from the IRQ when the scalar load/store command hits in the cache and a portion of one of the addresses of the one

or more addresses in the scalar load/store command does not match one or more partial addresses in the FOQ;

~~wherein one or more of the plurality of cache controllers includes an Force Order Queue (FOQ),~~ wherein the scalar load/store command is added to the FOQ when the scalar load/store command misses in the cache, and

wherein one or more lines in the cache are allocated for cache line replacement when the scalar load/store command is added to the FOQ and the address for the cache line does not match a partial address in the FOQ.

30. (Currently Amended) The scalar processor of claim 29, wherein ~~the cache controller with~~ the FOQ includes a FOQ index array.

31. (Previously Presented) The scalar processor of claim 29, wherein the FOQ is divided logically into a first and second queue, wherein the first queue manages requests to memory and the second queue manages data to be stored to cache.

32. (Previously Presented) The scalar processor of claim 22, wherein the FOQ is divided logically into a first and second queue, wherein the first queue manages requests to memory and the second queue manages data to be stored to cache.